Mobile Intel® Pentium® 4 Processor Supporting Hyper-Threading Technology† on 90 nm Process Technology

Specification Update

October 2005

Notice: The Mobile Intel® Pentium® 4 Processor may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are documented in this Specification Update.

Document Number: 302441-013

Contents

Revision History

Rev.	Revisions	Date
-001	Initial release	June 2004
-002	Revisions include: • Added Errata Q30-Q34 • Updated title of Errata Q21	August 2004
-003	Revisions include: • Added Errata Q35-Q45	September 2004
-004	Revisions include: • Added Errata Q46-Q51	October 2004
-005	Revisions include: • Added Errata Q52 • Updated Errata Q49 • Added Specification Change Q1 for T73: Deep Sleep PLL Lock Latency	November 2004
-006	Revisions include: • Updated Errata Q2 • Updated Specification Change Q1 • Added Errata Q53-54	December 2004
-007	Revisions include: • Updated Summary Tables of Changes	January 2005
-008	Revisions include: • Updated Summary Tables of Changes • Added Erratum Q55	February 2005
-009	• Updated Table 1: Processor Identification Information • Updated Errata Q29 in Summary Tables of Changes • Added Erratum Q56 • Added Specification Clarification Q1	May 2005
-010	• Added Erratum Q57	July 2005
-011	• Added Erratum Q58	August 2005
-012	• Updated Related Documents Table • Added Erratum Q59-60	September 2005
-013	• Added Erratum Q61-62	October 2005

§

Preface

This document is an update to the specifications contained in the documents listed in the following Affected Documents/Related Documents table. It is a compilation of device and document errata and specification clarifications and changes, and is intended for hardware system manufacturers and for software developers of applications, operating system, and tools.

Information types defined in the Nomenclature section of this document are consolidated into this update document and are no longer published in other documents. This document may also contain information that has not been previously published.

It is intended for hardware system manufacturers and software developers of applications, operating systems, or tools. It contains S-Specs, Errata, Documentation Changes, Specification Clarifications and Specification Changes.

Affected Documents

Document Title	Document Number
Mobile Intel® Pentium® 4 Processor supporting Hyper-Threading Technology on 90-nm process technology Datasheet	302424

NOTE: Hyper-Threading Technology requires a computer system with a Mobile Intel® Pentium® 4 processor supporting HT Technology and a Hyper-Threading Technology enabled chipset, BIOS and operating system. Performance will vary depending on the specific hardware and software you use. See <<http://www.intel.com/info/hyperthreading/>> for more information including details on which processors support HT Technology.

Related Documents

Document Title	Document Number
IA-32 Intel® Architecture Software Developer's Manual Volume 1: Basic Architecture	253665
IA-32 Intel® Architecture Software Developer's Manual Volume 2A: Instruction Set Reference Manual A–M	253666
IA-32 Intel® Architecture Software Developer's Manual Volume 2B: Instruction Set Reference Manual, N–Z	253667
IA-32 Intel® Architecture Software Developer's Manual Volume 3: System Programming Guide	253668
IA-32 Intel® Architecture Optimization Reference Manual	248966

Nomenclature

S-Spec Number is a five-digit code used to identify products. Products are differentiated by their unique characteristics, e.g., core speed, L2 cache size, package type, etc. as described in the processor identification information table. Care should be taken to read all notes associated with each S-Spec number

Errata are design defects or errors. Errata may cause the Mobile Intel® Pentium® 4 Processor's behavior to deviate from published specifications. Hardware and software designed to be used with any given stepping must assume that all errata documented for that stepping are present on all devices.

Specification Changes are modifications to the current published specifications. These changes will be incorporated in the next release of the specifications.

Specification Clarifications describe a specification in greater detail or further highlight a specification's impact to a complex design situation. These clarifications will be incorporated in the next release of the specifications.

Documentation Changes include typos, errors, or omissions from the current published specifications. These changes will be incorporated in the next release of the specifications.

Summary Tables of Changes

The following table indicates the Specification Changes, Errata, Specification Clarifications or Documentation Changes which apply to the listed Mobile Intel® Pentium® 4 Processor supporting Hyper-Threading Technology on 90-nm process technology steppings. Intel intends to fix some of the errata in a future stepping of the component and to account for the other outstanding issues through documentation or Specification Changes as noted. This table uses the following notations:

Codes Used in Summary Table

Stepping

X:	Erratum, Specification Change or Clarification that applies to this stepping.
(No mark) or (Blank Box):	This erratum is fixed in listed stepping or specification change does not apply to listed stepping.

Status

Doc:	Document change or update that will be implemented.
PlanFix:	This erratum may be fixed in a future stepping of the product.
Fixed:	This erratum has been previously fixed.
NoFix:	There are no plans to fix this erratum.

Row

Shaded:	This item is either new or modified from the previous version of the document.

Each Specification Update item will be prefixed with a capital letter to distinguish the product. The key below details the letters that are used in Intel's microprocessor Specification Updates:

A = Intel® Pentium® II processor
B = Mobile Intel® Pentium® II processor
C = Intel® Celeron® processor
D = Intel® Pentium® II Xeon® processor
E = Intel® Pentium® III processor
F = Intel® Pentium® processor Extreme Edition and Intel® Pentium® D processor
G = Intel® Pentium® III Xeon® processor
H = Mobile Intel® Celeron® processor at 466 MHz, 433 MHz, 400 MHz, 366 MHz, 333 MHz, 300 MHz, and 266 MHz
J = 64-bit Intel® Xeon® processor MP with 1-MB L2 cache

K = Mobile Intel® Pentium® III Processor – M
L = Intel® Celeron® D processor
M =Mobile Intel® Celeron® processor
N = Intel® Pentium® 4 processor
O = Intel® Xeon® processor MP
P = Intel® Xeon® processor
Q = Mobile Intel® Pentium® 4 processor supporting Hyper-Threading Technology on 90-nm technology process
R = Intel® Pentium® 4 processor on 90 nm process
S = 64-bit Intel® Xeon® processor with 800 MHz system bus (1-MB and 2-MB L2 cache versions)
T = Mobile Intel® Pentium® 4 processor – M
U = 64-bit Intel® Xeon® processor MP with up to 8-MB L3 cache
V = Mobile Intel® Celeron® processor on .13 Micron process in Micro-FCPGA Package
W = Intel® Celeron® M processor`
X = Intel® Pentium® M processor on 90nm process with 2-MB L2 cache
Y = Intel® Pentium® M processor
Z = Mobile Intel® Pentium® 4 processor with 533 MHz system bus

The Specification Updates for the Pentium processor, Pentium Pro processor, and other Intel products do not use this convention.

NO.	D0	E0	Plans	ERRATA
Q1	X	X	NoFix	Transaction is not retried after BINIT#
Q2	X	X	NoFix	Invalid opcode 0FFFH requires a ModRM byte
Q3	X	X	NoFix	Processor may hang due to Speculative Page Walks to Non-Existent System Memory
Q4	X	X	NoFix	Memory type of the load lock different from its corresponding store unlock
Q5	X	X	NoFix	Machine check architecture error reporting and recovery may not work as expected
Q6	X	X	NoFix	Debug mechanisms may not function as expected
Q7	X	X	NoFix	Cascading of performance counters does not work correctly when forced overflow is enabled
Q8	X	X	NoFix	EMON event counting of X87 loads may not work as expected
Q9	X	X	NoFix	System bus interrupt messages without data and which receive a HardFailure response may hang the processor
Q10	X	X	NoFix	The Processor Signals Page-Fault Exception (#PF) Instead of Alignment Check Exception (#AC) on an Unlocked CMPXCHG8B Instruction
Q11	X	X	NoFix	FSW may not be completely restored after page fault on FRSTOR or FLDENV instructions
Q12	X	X	NoFix	Processor Issues Inconsistent Transaction Size Attributes for Locked Operation

NO.	D0	E0	Plans	ERRATA
Q13	X	X	NoFix	When the processor is in the System Management Mode (SMM), Debug Registers may be fully writeable
Q14	X	X	NoFix	Shutdown and IERR# may result due to a Machine Check Exception on a Hyper-Threading technology enabled processor
Q15	X	X	NoFix	Processor may hang under certain frequencies and 12.5% STPCLK# duty cycle
Q16	X	X	NoFix	System may hang if a fatal cache error causes Bus Write Line (BWL) transaction to occur to the same cache line address as an outstanding Bus Read Line (BRL) or Bus Read-Invalidate Line (BRIL)
Q17	X	X	NoFix	A write to APIC Task Priority Register (TPR) that lowers priority may seem to have not occurred
Q18	X	X	NoFix	ITP cannot continue Single Step Execution after the first breakpoint
Q19	X	X	NoFix	Parity Error in the L1 Cache may cause the processor to hang
Q20	X	X	NoFix	A 16-bit Address Wrap Resulting from a Near Branch (Jump or Call) May Cause an Incorrect Address to Be Reported to the #GP Exception Handler
Q21	X		Fixed	Locks and SMC Detection May Cause the Processor to Temporarily Hang
Q22	X		PlanFix	Some Front Side Bus I/O Specifications are not Met
Q23	X		PlanFix	Incorrect Physical Address Size Returned by CPUID Instruction
Q24	X	X	NoFix	Incorrect Debug Exception (#DB) May Occur When a Data Breakpoint is set on an FP Instruction
Q25	X	X	NoFix	xAPIC May Not Report Some Illegal Vector Errors
Q26	X		PlanFix	Incorrect Duty Cycle is Chosen when On-Demand Clock Modulation is Enabled in a Processor Supporting Hyper-Threading Technology
Q27	X	X	NoFix	Memory Aliasing of Pages as Uncacheable Memory Type and Write Back (WB) May Hang the System
Q28	X		PlanFix	Enabling No-Eviction Mode (NEM) May Prevent the Operation of the Second Logical Processor in a Hyper-Threading Technology Enabled Processor
Q29	X	X	NoFix	Interactions Between the Instruction Translation Lookaside Buffer (ITLB) and the Instruction Streaming Buffer May Cause Unpredictable Software Behavior
Q30	X		PlanFix	STPCLK# Signal Assertion under Certain Conditions May Cause a System Hang
Q31	X		PlanFix	Sequence of Locked Operations Can Cause Two Threads to Receive Stale Data and Cause Application Hang
Q32	X	X	NoFix	Using STPCLK and Executing Code From Very Slow Memory Could Lead to a System Hang
Q33	X	X	NoFix	Processor Provides a 4-Byte Store Unlock After an 8-Byte Load Lock
Q34	X	X	NoFix	Machine check architecture error reporting and recovery may not work as expected

NO.	D0	E0	Plans	ERRATA
Q35	X		Fixed	EFLAGS.RF May be Incorrectly Set After an IRET Instruction
Q36	X		Fixed	Writing the Echo TPR Disable Bit in IA32_MISC_ENABLE May Cause a #GP Fault
Q37	X		Fixed	Incorrect Access Controls to MSR_LASTBRANCH_0_FROM_LIP MSR Registers
Q38	X	X	NoFix	Recursive Page Walks May Cause a System Hang
Q39	X		Fixed	WRMSR to bit[0] of IA32_MISC_ENABLE Register Changes Only One Logical Processor on a Hyper-Threading Technology Enabled Processor
Q40	X	X	NoFix	Data Breakpoints on the High Half of a Floating Point Line Split may not be Captured
Q41	X	X	NoFix	Machine Check Exceptions May not Update Last-Exception Record MSRs (LERs)
Q42	X	X	NoFix	MOV CR3 Performs Incorrect Reserved Bit Checking When in PAE Paging
Q43	X	X	NoFix	Stores to Page Tables May Not Be Visible to Pagewalks for Subsequent Loads Without Serializing or Invalidating the Page Table Entry
Q44	X		PlanFix	A Split Store Memory Access May Miss a Data Breakpoint
Q45	X	X	PlanFix	Execution of IRET or INTn Instructions May Cause Unexpected System Behavior
Q46	X	X	NoFix	Checking of Page Table Base Address May Not Match the Address Bit Width Supported by the Platform
Q47	X	X	NoFix	The IA32_MCi_STATUS MSR May Improperly Indicate that Additional MCA Information Has Been Captured
Q48	X	X	NoFix	With TF (Trap Flag) Asserted, FP Instruction That Triggers an Unmasked FP Exception May Take Single Step Trap Before Retirement of Instruction
Q49	X		Fixed	MCA Corrected Memory Hierarchy Error Counter May Not Increment Correctly
Q50	X	X	NoFix	BTS(Branch Trace Store) and PEBS(Precise Event Based Sampling) May Update Memory outside the BTS/PEBS Buffer
Q51	X		PlanFix	Processor May Hang When Resuming from Deep Sleep State
Q52	X	X	No Fix	Memory Ordering Failure May Occur with Snoop Filtering Third Party Agents after Issuing and Completing a BWIL (Bus Write Invalidate Line) or BLW (Bus Locked Write) Transaction
Q53	X	X	No Fix	Control Register 2 (CR2) Can be Updated during a REP MOVS/STOS Instruction with Fast Strings Enabled
Q54	X	X	PlanFix	TPR (Task Priority Register) Updates during Voltage Transitions of Power Management Events May Cause a System Hang
Q55	X	X	PlanFix	Running in SMM (System Management Mode) and L1 Data Cache Adaptive Mode May Cause Unexpected System Behavior when SMRAM is Mapped to Cacheable Memory
Q56	X	X	No Fix	Voltage and Frequency Transition May Not Occur if a Voltage Transition is Interrupted by a Warm Reset

NO.	D0	E0	Plans	ERRATA
Q57	X	X	Plan Fix	It Is Possible That Two Specific Invalid Opcodes May Cause Unexpected Memory Accesses
Q58	X	X	No Fix	At Core-to-Bus Ratios of 16:1 and above Defer Reply Transactions with Non-Zero REQb Values May Cause a Front Side Bus Stall
Q59	X	X	No Fix	The Processor May Issue Front Side Bus Transactions up to 6 Clocks after RESET# is Asserted
Q60	X	X	No Fix	Front Side Bus Machine Checks May be Reported as a Result of On-Going Transactions during Warm Reset
Q61	X	X	No Fix	Writing the Local Vector Table (LVT) when an Interrupt is Pending May Cause an Unexpected Interrupt
Q62	X	X	No Fix	The Processor May Issue Multiple Code Fetches to the Same Cache Line for Systems with Slow Memory

Number	SPECIFICATION CHANGES
Q1	Update to Processor T73: Deep Sleep PLL Lock Latency

Number	SPECIFICATION CLARIFICATIONS
Q1	Specification Clarification with Respect to Time-stamp Counter

Number	DOCUMENTATION CHANGES
	There are no Documentation Changes in this Specification Update revision

§

Identification Information

The processor can be identified by the following values:

Family[1]	Model[2]
1111b	0011b

NOTES:

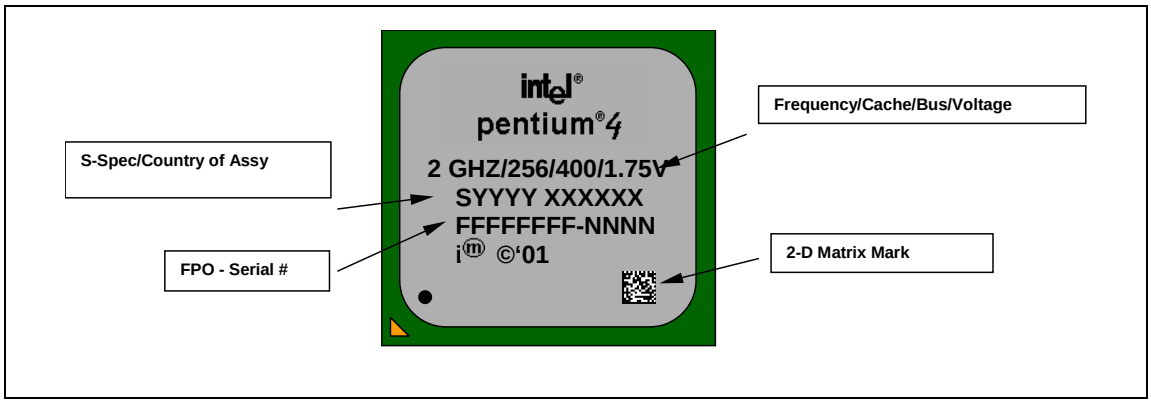
1. The Family corresponds to bits [11:8] of the EDX register after RESET, bits [11:8] of the EAX register after the CPUID instruction is executed with a 1 in the EAX register, and the generation field of the Device ID register accessible through Boundary Scan.
2. The Model corresponds to bits [7:4] of the EDX register after RESET, bits [7:4] of the EAX register after the CPUID instruction is executed with a 1 in the EAX register, and the model field of the Device ID register accessible through Boundary Scan.

Table 1. Mobile Intel® Pentium® 4 Processor Supporting Hyper-Threading Technology on 90 nm Process Technology Identification Information

S-Spec	Core Stepping	CPU Signature	Core Freq (GHz)	Data Bus Freq (MHz)	L2 Cache Size	Package and Revision	Notes
SL7DS	D0	0F34h	2.80	533	1 MB	478-pin micro-PGA with 35.0 x 35.0 mm FC-mPGA4 package	
SL7DT	D0	0F34h	3.06	533	1 MB	478-pin micro-PGA with 35.0 x 35.0 mm FC-mPGA4 package	
SL7DU	D0	0F34h	3.20	533	1 MB	478-pin micro-PGA with 35.0 x 35.0 mm FC-mPGA4 package	
SL7N8	E0	0F41h	2.80	533	1 MB	478-pin micro-PGA with 35.0 x 35.0 mm FC-mPGA4 package	
SL7NA	E0	0F41h	3.06	533	1 MB	478-pin micro-PGA with 35.0 x 35.0 mm FC-mPGA4 package	
SL7NB	E0	0F41h	3.20	533	1 MB	478-pin micro-PGA with 35.0 x 35.0 mm FC-mPGA4 package	
SL7X5	E0	0F41h	3.33	533	1 MB	478-pin micro-PGA with 35.0 x 35.0 mm FC-mPGA4 package	
SL7NC	E0	0F41h	3.46	533	1 MB	478-pin micro-PGA with 35.0 x 35.0 mm FC-mPGA4 package	

Component Marking Information

Figure 1. Mobile Intel® Pentium® 4 Processor Supporting Hyper-Threading Technology on 90 nm Process Technology Package Markings



§

Errata

Q1. Transaction Is Not Retired after BINIT#

Problem: If the first transaction of a locked sequence receives a HITM# and DEFER# during the snoop phase it should be retried and the locked sequence restarted. However, if BINIT# is also asserted during this transaction, the transaction will not be retried.

Implication: When this erratum occurs, locked transactions will not be retried.

Workaround: None identified.

Status: For the steppings affected, see the *Summary Tables of Changes*.

Q2. Invalid Opcode 0FFFh Requires a ModRM Byte

Problem: Some invalid opcodes require a ModRM byte and other following bytes, while others do not. The invalid opcode 0FFFh did not require a ModRM in previous generation microprocessors such as Pentium II or Pentium III processors, but it is required in the Intęl Pentium 4 processors and the Mobile Intel® Pentium® 4 processor supporting Hyper-Threading Technology on 90-nm technology process processor

Implication: The use of an invalid opcode 0FFFh without the ModRM byte may result **in a page or limit fault on the** Prescott processor. When this erratum occurs, locked transactions will not be retried.

Workaround: To avoid this erratum use ModRM byte with invalid 0FFFh opcode.

Status: For the steppings affected, see the *Summary Tables of Changes*.

Q3. Processor May Hang Due to Speculative Page Walks to Nonexistent System Memory

Problem: A load operation issued speculatively by the processor that misses the Data Translation Lookaside Buffer (DTLB) results in a page walk. A branch instruction older than the load retires so that this load operation is now in the mispredicted branch path. Due to an internal boundary condition, in some instances the load is not canceled before the page walk is issued.

The Page Miss Handler (PMH) starts a speculative page-walk for the Load and issues a cacheable load of the Page Directory Entry (PDE). This PDE load returns data that points to a page table entry in uncacheable (UC) memory. The PMH issues the PTE Load to UC space, which is issued on the Front Side Bus. No response comes back for this load PTE operation since the address is pointing to system memory, which does not exist.

This load to non-existent system memory causes the processor to hang because other bus requests are queued up behind this UC PTE load, which never gets a response. If the load was accessing valid system memory, the speculative page-walk would successfully complete and the processor would continue to make forward progress.

Implication: Processor may hang due to speculative page walks to non-existent system memory.

Workaround: Page directories and page tables in UC memory space must point to system memory that exists.

Status: For the steppings affected, see the *Summary Tables of Changes*.

Q4. Memory Type of the Load Lock Different from Its Corresponding Store Unlock

Problem: A use-once protocol is employed to ensure that the processor in a multi-agent system may access data that is loaded into its cache on a Read-for-Ownership operation at least once before it is snooped out by another agent. This protocol is necessary to avoid a multi-agent livelock scenario in which the processor cannot gain ownership of a line and modify it before that data is snooped out by another agent. In the case of this erratum, split load lock instructions incorrectly trigger the use-once protocol. A load lock operation accesses data that splits across a page boundary with both pages of WB memory type. The use-once protocol activates and the memory type for the split halves get forced to UC. Since use-once does not apply to stores, the store unlock instructions go out as WB memory type. The full sequence on the bus is: locked partial read (UC), partial read (UC), partial write (WB), locked partial write (WB). The use-once protocol should not be applied to load locks.

Implication: When this erratum occurs, the memory type of the load lock will be different than the memory type of the store unlock operation. This behavior (Load Locks and Store Unlocks having different memory types) does not however introduce any functional failures such as system hangs or memory corruption.

Workaround: None identified.

Status: For the steppings affected, see the *Summary Tables of Changes*.

Q5. Machine Check Architecture Error Reporting and Recovery May Not Work As Expected

Problem: When the processor detects errors it should attempt to report and/or recover from the error. In the situations described below, the processor does not report and/or recover from the error(s) as intended.

When a transaction is deferred during the snoop phase and subsequently receives a Hard Failure response, the transaction should be removed from the bus queue so that the processor may proceed. Instead, the transaction is not properly removed from the bus queue, the bus queue is blocked, and the processor will hang.

When a hardware prefetch results in an uncorrectable tag error in the L2 cache, MC0_STATUS.UNCOR and MC0_STATUS.PCC are set but no Machine Check Exception (MCE) is signaled. No data loss or corruption occurs because the data being prefetched has not been used. If the data location with the uncorrectable tag error is subsequently accessed, an MCE will occur. However, upon this MCE, or any other subsequent MCE, the information for that error will not be logged because MC0_STATUS.UNCOR has already been set and the MCA status registers will not contain information about the error which caused the MCE assertion but instead will contain information about the prefetch error event.

When the reporting of errors is disabled for Machine Check Architecture (MCA) Bank 2 by setting all MC2_CTL register bits to 0, uncorrectable errors should be logged in the IA32_MC2_STATUS register but no machine-check exception should be generated. Uncorrectable loads on bank 2, which would normally be logged in the IA32_MC2_STATUS register, are not logged.

When one half of a 64 byte instruction fetch from the L2 cache has an uncorrectable error and the other 32 byte half of the same fetch from the L2 cache has a correctable error, the processor will attempt to

correct the correctable error but cannot proceed due to the uncorrectable error. When this occurs the processor will hang.

When an L1 cache parity error occurs, the cache controller logic should write the physical address of the data memory location that produced that error into the IA32_MC1_ADDR REGISTER (MC1_ADDR). In some instances of a parity error on a load operation that hits the L1 cache, however, the cache controller logic may write the physical address from a subsequent load or store operation into the IA32_MC1_ADDR register.

When an error exists in the tag field of a cache line such that a request for ownership (RFO) issued by the processor hits multiple tag fields in the L2 cache (the correct tag and the tag with the error) and the accessed data also has a correctable error, the processor will correctly log the multiple tag match error but will hang when attempting to execute the machine check exception handler.

If a memory access receives a machine check error on both 64 byte halves of a 128-byte L2 cache sector, the IA32_MC0_STATUS register records this event as multiple errors, i.e., the valid error bit and the overflow error bit are both set indicating that a machine check error occurred while the results of a previous error were in the error-reporting bank. The IA32_MC1_STATUS register should also record this event as multiple errors but instead records this event as only one correctable error.

The overflow bit should be set to indicate when more than one error has occurred. The overflow bit being set indicates that more than one error has occurred. Because of this erratum, if any further errors occur, the MCA overflow bit will not be updated, thereby incorrectly indicating only one error has been received.

If an I/O instruction (IN, INS, REP INS, OUT, OUTS, or REP OUTS) is being executed, and if the data for this instruction becomes corrupted, the processor will signal a Machine Check Exception (MCE). If the instruction is directed at a device that is powered down, the processor may also receive an assertion of SMI#. Since MCEs have higher priority, the processor will call the MCE handler, and the SMI# assertion will remain pending. However, while attempting to execute the first instruction of the MCE handler, the SMI# will be recognized and the processor will attempt to execute the SMM handler. If the SMM handler is successfully completed, it will attempt to restart the I/O instruction, but will not have the correct machine state due to the call to the MCE handler. This can lead to failure of the restart and shutdown of the processor.

If PWRGOOD is de-asserted during a RESET# assertion causing internal glitches, the MCA registers may latch invalid information.

If RESET# is asserted, then de-asserted, and reasserted, before the processor has cleared the MCA registers, then the information in the MCA registers may not be reliable, regardless of the state or state transitions of PWRGOOD.

If MCERR# is asserted by one processor and observed by another processor, the observing processor does not log the assertion of MCERR#. The Machine Check Exception (MCE) handler called upon assertion of MCERR# will not have any way to determine the cause of the MCE.

The Overflow Error bit (bit 62) in the IA32_MC0_STATUS register indicates, when set, that a machine check error occurred while the results of a previous error were still in the error reporting bank (i.e. The Valid bit was set when the new error occurred). If an uncorrectable error is logged in the error-reporting bank and another error occurs, the overflow bit will not be set.

The MCA Error Code field of the IA32_MC0_STATUS register gets written by a different mechanism than the rest of the register. For uncorrectable errors, the other fields in the IA32_MC0_STATUS register are only updated by the first error. Any further errors that are detected will update the MCA Error Code field without updating the rest of the register, thereby leaving the IA32_MC0_STATUS register with stale information.

When a speculative load operation hits the L2 cache and receives a correctable error, the IA32_MC1_Status Register may be updated with incorrect information. The IA32_MC1_Status Register should not be updated for speculative loads.

The processor should only log the address for L1 parity errors in the IA32_MC1_Status register if a valid address is available. If a valid address is not available, the Address Valid bit in the IA32_MC1_Status register should not be set. In instances where an L1 parity error occurs and the address is not available because the linear to physical address translation is not complete or an internal resource conflict has occurred, the Address Valid bit is incorrectly set.

The processor may hang when an instruction code fetch receives a hard failure response from the Front Side Bus. This occurs because the bus control logic does not return data to the core, leaving the processor empty. IA32_MC0_STATUS MSR does indicate that a hard fail response occurred.

The processor may hang when the following events occur and the machine check exception is enabled, CR4.MCE=1. A processor that has its STPCLK# pin asserted will internally enter the Stop Grant State and finally issue a Stop Grant Acknowledge special cycle to the bus. If an uncorrectable error is generated during the Stop Grant process it is possible for the Stop Grant special cycle to be issued to the bus before the processor vectors to the machine check handler. Once the chipset receives its last Stop Grant special cycle it is allowed to ignore any bus activity from the processors. As a result, processor accesses to the machine check handler may not be acknowledged, resulting in a processor hang.

Implication: The processor is unable to correctly report and/or recover from certain errors

Workaround: None identified.

Status: For the steppings affected, see the *Summary Tables of Changes*.

Q6. Debug Mechanisms May Not Function As Expected

Problem: Certain debug mechanisms may not function as expected on the processor. The cases are as follows:

- When the following conditions occur: 1) An FLD instruction signals a stack overflow or underflow, 2) the FLD instruction splits a page-boundary or a 64 byte cache line boundary, 3) the instruction matches a Debug Register on the high page or cache line respectively, and 4) the FLD has a stack fault and a memory fault on a split access, the processor will only signal the stack fault and the debug exception will not be taken.
- When a data breakpoint is set on the ninth and/or tenth byte(s) of a floating point store using the Extended Real data type, and an unmasked floating point exception occurs on the store, the break point will not be captured.
- When any instruction has multiple debug register matches, and any one of those debug registers is enabled in DR7, all of the matches should be reported in DR6 when the processor goes to the debug handler. This is not true during a REP instruction. As an example, during execution of a REP MOVSW instruction the first iteration a load matches DR0 and DR2 and sets DR6 as FFFF0FF5h. On a subsequent iteration of the instruction, a load matches only DR0. The DR6 register is expected to still contain FFFF0FF5h, but the processor will update DR6 to FFFF0FF1h.
- A data breakpoint that is set on a load to uncacheable memory may be ignored due to an internal segment register access conflict. In this case the system will continue to execute instructions, bypassing the intended breakpoint. Avoiding having instructions that access segment descriptor registers, e.g., LGDT, LIDT close to the UC load, and avoiding serialized instructions before the UC load will reduce the occurrence of this erratum.

Implication: Certain debug mechanisms do not function as expected on the processor.

Workaround: None identified.

Status: For the steppings affected, see the *Summary Tables of Changes*.

Q7. Cascading Of Performance Counters Does Not Work Correctly When Forced Overflow Is Enabled

Problem: The performance counters are organized into pairs. When the CASCADE bit of the Counter Configuration Control Register (CCCR) is set, a counter that overflows will continue to count in the other counter of the pair. The FORCE_OVF bit forces the counters to overflow on every non-zero increment. When the FORCE_OVF bit is set, the counter overflow bit will be set but the counter no longer cascades.

Implication: The performance counters do not cascade when the FORCE_OVF bit is set.

Workaround: None identified.

Status: For the steppings affected, see the *Summary Tables of Changes*.

Q8. EMON Event Counting of x87 Loads May Not Work As Expected

Problem: If a performance counter is set to count x87 loads and floating-point exceptions are unmasked, the FPU Operand (Data) Pointer (FDP) may become corrupted.

Implication: When this erratum occurs, FPU Operand (Data) Pointer (FDP) may become corrupted.

Workaround: This erratum will not occur with floating point exceptions masked. If floating-point exceptions are unmasked, then performance counting of x87 loads should be disabled.

Status: For the steppings affected, see the *Summary Tables of Changes*.

Q9. System Bus Interrupt Messages Without Data and Which Receive a HardFailure Response May Hang the Processor

Problem: When a System Bus agent (processor or chipset) issues an interrupt transaction without data onto the System Bus, and the transaction receives a HardFailure response, a potential processor hang can occur. The processor, which generates an inter-processor interrupt (IPI) that receives HardFailure response, will still log the MCA error event cause as HardFailure, even if the APIC causes a hang. Other processors, which are true targets of the IPI, will also hang on hardfailure-without-data, but will not record an MCA HardFailure event as a cause. If a HardFailure response occurs on a System Bus interrupt message with data, the APIC will complete the operation so as not to hang the processor.

Implication: The processor may hang.

Workaround: None identified.

Status: For the steppings affected, see the *Summary Tables of Changes*.

Q10. The Processor Signals Page-Fault Exception (#PF) Instead of Alignment Check Exception (#AC) on an Unlocked CMPXCHG8B Instruction

Problem: If a Page-Fault Exception (#PF) and Alignment Check Exception (#AC) both occur for an unlocked CMPXCHG8B instruction, then #PF will be flagged.

Implication: Software that depends on the Alignment Check Exception (#AC) before the Page-Fault Exception (#PF) will be affected since #PF is signaled in this case.

Workaround: Remove the software's dependency on #AC having precedence over #PF. Alternately, correct the page fault in the page fault handler and then restart the faulting instruction.

Status: For the steppings affected, see the *Summary Tables of Changes*.

Q11. FSW May not be Completely Restored after Page Fault on FRSTOR or FLDENV Instructions

Problem: If the FPU operating environment or FPU state (operating environment and register stack) being loaded by an FLDENV or FRSTOR instruction wraps around a 64-Kbyte or 4-Gbyte boundary and a page fault (#PF) or segment limit fault (#GP or #SS) occurs on the instruction near the wrap boundary, the upper byte of the FPU status word (FSW) might not be restored. If the fault handler does not restart program execution at the faulting instruction, stale data may exist in the FSW.

Implication: When this erratum occurs, stale data will exist in the FSW.

Workaround: Ensure that the FPU operating environment and FPU state do not cross 64-Kbyte or 4-Gbyte boundaries. Alternately, ensure that the page fault handler restarts program execution at the faulting instruction after correcting the paging problem.

Status: For the steppings affected, see the *Summary Tables of Changes*.

Q12. Processor Issues Inconsistent Transaction Size Attributes for Locked Operation

Problem: When the processor is in the Page Address Extension (PAE) mode and detects the need to set the Access and/or Dirty bits in the page directory or page table entries, the processor sends an 8 byte load lock onto the System Bus. A subsequent 8 byte store unlock is expected, but instead a 4 byte store unlock occurs. Correct data is provided since only the lower bytes change, however external logic monitoring the data transfer may be expecting an 8-byte store unlock.

Implication: No known commercially available chipsets are affected by this erratum.

Workaround: None identified.

Status: For the steppings affected, see the *Summary Tables of Changes*.

Q13. When the Processor is in the System Management Mode (SMM), Debug Registers May Be Fully Writeable

Problem: When in System Management Mode (SMM), the processor executes code and stores data in the SMRAM space. When the processor is in this mode and writes are made to DR6 and DR7, the processor should block writes to the reserved bit locations. Due to this erratum, the processor may not block these writes. This may result in invalid data in the reserved bit locations.

Implication: Reserved bit locations within DR6 and DR7 may become invalid.

Workaround: Software may perform a read/modify/write when writing to DR6 and DR7 to ensure that the value in the reserved bits are maintained.

Status: For the steppings affected, see the *Summary Tables of Changes*.

Q14. Shutdown and IERR# May Result Due to a Machine Check Exception on a Hyper-Threading Technology Enabled Processor

Problem: When a Machine Check Exception (MCE) occurs due to an internal error, both logical processors on a Hyper Threading technology enabled processor normally vector to the MCE handler. However, if one of the logical processors is in the "Wait for SIPI" state, that logical processor will not have a MCE handler and will shut down and assert IERR#.

Implication: A processor with a logical processor in the "Wait for SIPI" state will shut down when an MCE occurs on the other thread.

Workaround: None identified.

Status: For the steppings affected, see the *Summary Tables of Changes*.

Q15. Processor May Hang Under Certain Frequencies and 12.5% STPCLK# Duty Cycle

Problem: If a system de-asserts STPCLK# at a 12.5% duty cycle, and the processor is running below 2 GHz, and the processor thermal control circuit (TCC) on-demand clock modulation is active, the processor may hang. This erratum does not occur under the automatic mode of the TCC.

Implication: When this erratum occurs, the processor will hang.

Workaround: If use of the on-demand mode of the processor's TCC is desired in conjunction with STPCLK# modulation, then assure that STPCLK# is not asserted at a 12.5% duty cycle.

Status: For the steppings affected, see the *Summary Tables of Changes*.

Q16. System May Hang if a Fatal Cache Error Causes Bus Write Line (BWL) Transaction to Occur to the Same Cache Line Address as an Outstanding Bus Read Line (BRL) or Bus Read-Invalidate Line (BRIL)

Problem: A processor internal cache fatal data ECC error may cause the processor to issue a BWL transaction to the same cache line address as an outstanding BRL or BRIL. As it is not typical behavior for a single processor to have a BWL and a BRL/BRIL concurrently outstanding to the same address, this may represent an unexpected scenario to system logic within the chipset.

Implication: The processor may not be able to fully execute the machine check handler in response to the fatal cache error if system logic does not ensure forward progress on the System Bus under this scenario.

Workaround: System logic should ensure completion of the outstanding transactions. Note that during recovery from a fatal data ECC error, memory image coherency of the BWL with respect to BRL/BRIL transactions is not important. Forward progress is the primary requirement.

Status: For the steppings affected, see the *Summary Tables of Changes*.

Q17. A Write to APIC Task Priority Register (TPR) that Lowers Priority May Seem to Have Not Occurred

Problem: Uncacheable stores to the APIC space are handled in a non-synchronous way with respect to the speed at which instructions are retired. If an instruction that masks the interrupt flag e.g. CLI is executed soon after an uncacheable write to the TPR that lowers the APIC priority the interrupt masking operation may take effect before the actual priority has been lowered. This may cause interrupts whose priority is lower than the initial TPR but higher than the final TPR to not be serviced until the interrupt flag is finally cleared e.g. STI. Interrupts will remain pended and are not lost

Implication: This condition may allow interrupts to be accepted by the processor but may delay their service.

Workaround: This can be avoided by issuing a TPR Read after a TPR Write that lowers the TPR value. This will force the store to the APIC priority resolution logic before any subsequent instructions are executed. No commercial operating system is known to be impacted by this erratum.

Status: For the steppings affected, see the *Summary Tables of Changes*.

Q18. ITP Cannot Continue Single Step Execution after the First Breakpoint

Problem: ITP will not continue in single step execution after the first software breakpoint. ITP is unable to reset the Resume Flag (RF) bit in the EFLAGS Register.

Implication: The processor repeatedly breaks at the instruction breakpoint address instead of single stepping.

Workaround: Execution after the break will continue if DR7 bit 1 (Global Breakpoint Enable) is manually cleared.

Status: For the steppings affected, see the *Summary Tables of Changes*.

Q19. Parity Error in the L1 Cache May Cause the Processor to Hang

Problem: If a locked operation accesses a line in the L1 cache that has a parity error, it is possible that the processor may hang while trying to evict the line.

Implication: If this erratum occurs, it may result in a system hang. Intel has not observed this erratum with any commercially available software.

Workaround: None identified.

Status: For the steppings affected, see the *Summary Tables of Changes*.

Q20. A 16-bit Address Wrap Resulting from a Near Branch (Jump or Call) May Cause an Incorrect Address to Be Reported to the #GP Exception Handler

Problem: If a 16-bit application executes a branch instruction that causes an address wrap to a target address outside of the code segment, the address of the branch instruction should be provided to the general protection exception handler. It is possible that, as a result of this erratum, that the general protection handler may be called with the address of the branch target.

Implication: The 16-bit software environment which is affected by this erratum, will see that the address reported by the exception handler points to the target of the branch, rather than the address of the branch instruction.

Workaround: None identified.

Status: For the steppings affected, see the *Summary Tables of Changes*.

Q21. Locks and SMC Detection May Cause the Processor to Temporarily Hang

Problem: The processor may temporarily hang in a Hyper-Threading Technology (HT Technology) enabled system if one logical processor executes a synchronization loop that includes one or more locks and is waiting for release by the other logical processor. If the releasing logical processor is executing instructions that are within the detection range of the self modifying code (SMC) logic, then the processor may be locked in the synchronization loop until the arrival of an interrupt or other event.

Implication: If this erratum occurs in a Hyper-Threading Technology (HT Technology) enabled system, the application may temporarily stop making forward progress. Intel has not observed this erratum with any commercially available software.

Workaround: None identified.

Status: For the steppings affected, see the *Summary Tables of Changes.*

Q22. Some Front Side Bus I/O Specifications Are Not Met

Problem: The following front side bus I/O specifications are not met:

- The VIH(min) for the GTL+ signals is specified as GTLREF + (0.10 * VCC) [V].
- The VIH(min) for the Asynchronous GTL+ signals is specified as Vcc/2 + (0.10 * VCC) [V].
- Common Clock Output Valid Delay(min) is specified as -0.250 ns.
- Common Clock Input Setup Time is specified as 0.700 ns.
- Source Synchronous Input Setup Time to Strobe is specified as 0.150 ns.
- Source Synchronous Input Hold Time to Strobe is specified as 0.150 ns.

Implication: This erratum can cause functional failures depending upon system bus activity. It can manifest itself as data parity, address parity, and/or machine check errors.

Workaround: Due to this erratum, the system should meet the following voltage levels and processor timings:

- The $V_{IH(min)}$ for GTL+ signals is now GTLREF + (0.20 * V_{CC}) [V].
- The $V_{IH(min)}$ for the Asynchronous GTL+ signals is now Vcc/2 + (0.20 * V_{CC}) [V].
- Common Clock Output Valid Delay(min) is now -0.300 ns.
- Common Clock Input Setup Time is now 0.900 ns.
- Source Synchronous Input Setup Time to Strobe is now 0.350 ns.
- Source Synchronous Input Hold Time to Strobe is now 0.350 ns.

Status: For the steppings affected, see the *Summary Tables of Changes*.

Q23. Incorrect Physical Address Size Returned by CPUID Instruction

Problem: The CPUID instruction Function 80000008H (Extended Address Sizes Function) returns the address sizes supported by the processor in the EAX register. This Function returns an incorrect physical address size value of 40 bits. The correct physical address size is 36 bits.

Implication: Function 80000008H returns an incorrect physical address size value of 40 bits.

Workaround: None identified.

Status: For the steppings affected, see the *Summary Tables of Changes*.

Q24. Incorrect Debug Exception (#DB) May Occur When a Data Breakpoint Is Set on an FP Instruction

Problem: The default Microcode Floating Point Event Handler routine executes a series of loads to obtain data about the FP instruction that is causing the FP event. If a data breakpoint is set on the instruction causing the FP event, the load in the microcode routine will trigger the data breakpoint resulting in a debug exception.

Implication: An incorrect Debug Exception (#DB) may occur if data breakpoint is placed on an FP instruction. Intel has not observed this erratum with any commercially available software or system.

Workaround: None identified.

Status: For the steppings affected, see the *Summary Tables of Changes*.

Q25. xAPIC May Not Report Some Illegal Vector Errors

Problem: The local xAPIC has an Error Status Register, which records all errors. The bit 6 (the Receive Illegal Vector bit) of this register, is set when the local xAPIC detects an illegal vector in a received message. When an illegal vector error is received on the same internal clock that the error status register is being written (due to a previous error), bit 6 does not get set and illegal vector errors are not flagged.

Implication: The xAPIC may not report some Illegal Vector errors when they occur at approximately the same time as other xAPIC errors. The other xAPIC errors will continue to be reported.

Workaround: None Identified

Status: For the steppings affected, see the *Summary Tables of Changes*.

Q26. Incorrect Duty Cycle is Chosen when On-Demand Clock Modulation is Enabled in a Processor Supporting Hyper-Threading Technology

Problem: When a processor supporting Hyper-Threading Technology (HT Technology) enables On-Demand Clock Modulation on both logical processors, the processor is expected to select the lowest duty cycle of the two potentially different values. When one logical processor enters the AUTOHALT state, the duty cycle implemented should be unaffected by the halted logical processor. Due to this erratum, the duty cycle is incorrectly chosen to be the higher duty cycle of both logical processors.

Implication: Due to this erratum, higher duty cycle may be chosen when the On-Demand Clock Modulation is enabled on both logical processors.

Workaround: None identified.

Status: For the steppings affected, see the *Summary Tables of Changes*.

Q27. Memory Aliasing of Pages As Uncacheable Memory Type and Write Back (WB) May Hang the System

Problem: When a page is being accessed as either Uncacheable (UC) or Write Combining (WC) and WB, under certain bus and memory timing conditions, the system may loop in a continual sequence of UC fetch, implicit writeback, and Request For Ownership (RFO) retries.

Implication: This erratum has not been observed in any commercially available operating system or application. The aliasing of memory regions, a condition necessary for this erratum to occur, is documented as being unsupported in the *IA-32 Intel® Architecture Software Developer's Manual*, Volume 3, section 10.12.4, Programming the PAT. However, if this erratum occurs the system may hang.

Workaround: The pages should not be mapped as either UC or WC and WB at the same time.

Status: For the steppings affected, see the *Summary Tables of Changes*.

Q28. Enabling No-Eviction Mode (NEM) May Prevent the Operation of the Second Logical Processor in a Hyper-Threading Technology Enabled Processor

Problem: In an HT Technology enabled system, when NEM is enabled by setting Bit 0 of MSR 080h (IA32_BIOS_CACHE_AS_RAM), the second logical processor may fail to wake up from "Wait-for-SIPI" state.

Implication: In an HT Technology enabled system, the second logical processor may not respond to SIPI. The OS will continue to operate but only with a single logical processor.

Workaround: It is possible for the BIOS to contain a workaround for this erratum.

Status: For the steppings affected, see the *Summary Tables of Changes*.

Q29. Interactions Between the Instruction Translation Lookaside Buffer (ITLB) and the Instruction Streaming Buffer May Cause Unpredictable Software Behavior

Problem: Complex interactions within the instruction fetch / decode unit may make it possible for the processor to execute instructions from an internal streaming buffer containing stale or incorrect information.

Implication: When this erratum occurs, an incorrect instruction stream may be executed resulting in unpredictable software behavior.

Workaround: It is possible for the BIOS to contain a workaround for this erratum.

Status: For the steppings affected, see the *Summary Tables of Changes*.

Q30. STPCLK# Signal Assertion under Certain Conditions May Cause a System Hang

Problem: The assertion of STPCLK# signal before a logical processor awakens from the "Wait-for-SIPI" state for the first time, may cause a system hang. A processor supporting Hyper-Threading Technology may fail to initialize appropriately, and may not issue a Stop Grant Acknowledge special bus cycle in response to the second STPCLK# assertion.

Implication: When this erratum occurs in an HT Technology enabled system, it may cause a system hang.

Workaround: BIOS should initialize the second thread of the processor supporting Hyper-Threading Technology prior to STPCLK# assertion. Additionally, it is possible for the BIOS to contain a workaround for this erratum.

Status: For the steppings affected, see the *Summary Tables of Changes*.

Q31. Sequence of Locked Operations Can Cause Two Threads to Receive Stale Data and Cause Application Hang

Problem: While going through a sequence of locked operations, it is possible for the two threads to receive stale data. This is a violation of expected memory ordering rules and the application may hang.

Implication: When this erratum occurs in an HT Technology-enabled system, the application may hang.

Workaround: It is possible for the BIOS to contain a workaround for this erratum.

Status: For the steppings affected, see the *Summary Tables of Changes*.

Q32. Using STPCLK and Executing Code from Very Slow Memory Could Lead to a System Hang

Problem: The system may hang when the following conditions are met:

1. Periodic STPCLK mechanism is enabled via the chipset
2. Hyper-Threading Technology is enabled
3. One logical processor is waiting for an event (i.e. hardware interrupt)
4. The other logical processor executes code from very slow memory such that every code fetch is deferred long enough for the STPCLK to be re-asserted.

Implication: If this erratum occurs, the processor will go into and out of the sleep state without making forward progress, since the logical processor will not be able to service any pending event. This erratum has not been observed in any commercial platform running commercial software.

Workaround: None identified at this time.

Status: For the steppings affected, see the *Summary Tables of Changes*.

Q33. Processor Provides a 4-Byte Store Unlock after an 8 –Byte Load Lock

Problem: When the processor is in the Page Address Extension (PAE) mode and detects the need to set the Access and/or Dirty bits in the page directory or page table entries, the processor sends an 8 byte load lock onto the system bus. A subsequent 8 byte store unlock is expected, but instead a 4 byte store unlock occurs. Correct data is provided since only the lower bytes change, however external logic monitoring the data transfer may be expecting an 8 byte load lock.

Implication: No known commercially available chipsets are affected by this erratum.

Workaround: None identified at this time.

Status: For the steppings affected, see the *Summary Tables of Changes*.

Q34. Machine Check Architecture Error Reporting and Recovery May Not Work As Expected

Problem: When the processor detects errors it should attempt to report and/or recover from the error. In the situations described below, the processor does not report and/or recover from the error(s) as intended.

- When a transaction is deferred during the snoop phase and subsequently receives a Hard Failure response, the transaction should be removed from the bus queue so that the processor may proceed. Instead, the transaction is not properly removed from the bus queue, the bus queue is blocked, and the processor will hang.
- When a hardware prefetch results in an uncorrectable tag error in the L2 cache, MC0_STATUS.UNCOR and MC0_STATUS.PCC are set but no Machine Check Exception (MCE) is signaled. No data loss or corruption occurs because the data being prefetched has not been used. If the data location with the uncorrectable tag error is subsequently accessed, an MCE will occur. However, upon this MCE, or any other subsequent MCE, .the information for that error will not be logged because MC0_STATUS.UNCOR has already been set and the MCA status registers will not contain information about the error which caused the MCE assertion but instead will contain information about the prefetch error event.

- When the reporting of errors is disabled for Machine Check Architecture (MCA) Bank 2 by setting all MC2_CTL register bits to 0, uncorrectable errors should be logged in the IA32_MC2_STATUS register but no machine-check exception should be generated. Uncorrectable loads on bank 2, which would normally be logged in the IA32_MC2_STATUS register, are not logged.
- When one-half of a 64-byte instruction fetch from the L2 cache has an uncorrectable error and the other 32-byte half of the same fetch from the L2 cache has a correctable error, the processor will attempt to correct the correctable error but cannot proceed due to the uncorrectable error. When this occurs the processor will hang.
- When an L1 cache parity error occurs, the cache controller logic should write the physical address of the data memory location that produced that error into the IA32_MC1_ADDR REGISTER (MC1_ADDR). In some instances of a parity error on a load operation that hits the L1 cache, the cache controller logic may write the physical address from a subsequent load or store operation into the IA32_MC1_ADDR register.
- When an error exists in the tag field of a cache line such that a request for ownership (RFO) issued by the processor hits multiple tag fields in the L2 cache (the correct tag and the tag with the error) and the accessed data also has a correctable error, the processor will correctly log the multiple tag match error but will hang when attempting to execute the machine check exception handler.
- If a memory access receives a machine check error on both 64 byte halves of a 128-byte L2 cache sector, the IA32_MC0_STATUS register records this event as multiple errors, i.e., the valid error bit and the overflow error bit are both set indicating that a machine check error occurred while the results of a previous error were in the error-reporting bank. The IA32_MC1_STATUS register should also record this event as multiple errors but instead records this event as only one correctable error.
- The overflow bit should be set to indicate when more than one error has occurred. The overflow bit being set indicates that more than one error has occurred. Because of this erratum, if any further errors occur, the MCA overflow bit will not be updated, thereby incorrectly indicating only one error has been received.
- If an I/O instruction (IN, INS, REP INS, OUT, OUTS, or REP OUTS) is being executed, and if the data for this instruction becomes corrupted, the processor will signal a Machine Check Exception (MCE). If the instruction is directed at a device that is powered down, the processor may also receive an assertion of SMI#. Since MCEs have higher priority, the processor will call the MCE handler, and the SMI# assertion will remain pending. However, while attempting to execute the first instruction of the MCE handler, the SMI# will be recognized and the processor will attempt to execute the SMM handler. If the SMM handler is successfully completed, it will attempt to restart the I/O instruction, but will not have the correct machine state due to the call to the MCE handler. This can lead to failure of the restart and shutdown of the processor.
- If PWRGOOD is de-asserted during a RESET# assertion causing internal glitches, the MCA registers may latch invalid information.
- If RESET# is asserted, then de-asserted, and reasserted, before the processor has cleared the MCA registers, then the information in the MCA registers may not be reliable, regardless of the state or state transitions of PWRGOOD.
- If MCERR# is asserted by one processor and observed by another processor, the observing processor does not log the assertion of MCERR#. The Machine Check Exception (MCE) handler called upon assertion of MCERR# will not have any way to determine the cause of the MCE.
- The Overflow Error bit (bit 62) in the IA32_MC0_STATUS register indicates, when set, that a machine check error occurred while the results of a previous error were still in the error reporting bank (i.e. The Valid bit was set when the new error occurred). If an uncorrectable error is logged in the error-reporting bank and another error occurs, the overflow bit will not be set.
- The MCA Error Code field of the IA32_MC0_STATUS register gets written by a different mechanism than the rest of the register. For uncorrectable errors, the other fields in the IA32_MC0_STATUS register are only updated by the first error. Any further errors that are detected

will update the MCA Error Code field without updating the rest of the register, thereby leaving the IA32_MC0_STATUS register with stale information.

- When a speculative load operation hits the L2 cache and receives a correctable error, the IA32_MC1_Status Register may be updated with incorrect information. The IA32_MC1_Status Register should not be updated for speculative loads.
- The processor should only log the address for L1 parity errors in the IA32_MC1_Status register if a valid address is available. If a valid address is not available, the Address Valid bit in the IA32_MC1_Status register should not be set. In instances where an L1 parity error occurs and the address is not available because the linear to physical address translation is not complete or an internal resource conflict has occurred, the Address Valid bit is incorrectly set.
- The processor may hang when an instruction code fetch receives a hard failure response from the system bus. This occurs because the bus control logic does not return data to the core, leaving the processor empty. IA32_MC0_STATUS MSR does indicate that a hard fail response occurred.
- The processor may hang when the following events occur and the machine check exception is enabled, CR4.MCE=1. A processor that has its STPCLK# pin asserted will internally enter the Stop Grant State and finally issue a Stop Grant Acknowledge special cycle to the bus. If an uncorrectable error is generated during the Stop Grant process it is possible for the Stop Grant special cycle to be issued to the bus before the processor vectors to the machine check handler. Once the chipset receives its last Stop Grant special cycle it is allowed to ignore any bus activity from the processors. As a result, processor accesses to the machine check handler may not be acknowledged, resulting in a processor hang.

Implication: The processor is unable to correctly report and/or recover from certain errors.

Workaround: None identified.

Status: For the steppings affected, see the *Summary Tables of Changes*.

Q35. EFLAGS.RF May Be Incorrectly Set after an IRET Instruction

Problem: EFLAGS.RF is used to disable code breakpoints. After an IRET instruction, EFLAGS.RF may be incorrectly set or not set depending on its value right before the IRET instruction.

Implication: A code breakpoint may be missed or an additional code breakpoint may be taken on next instruction.

Workaround: It is possible for the BIOS to contain a workaround for this erratum.

Status: For the steppings affected, see the *Summary Tables of Changes*.

Q36. Writing the Echo TPR Disable Bit in IA32_MISC_ENABLE May Cause a #GP Fault

Problem: Writing a '1' to the Echo TPR disable bit (bit 23) in IA32_MISC_ENABLE may incorrectly cause a #GP fault.

Implication: A #GP fault may occur if the bit is set to a '1'.

Workaround: It is possible for the BIOS to contain a workaround for this erratum.

Status: For the steppings affected, see the *Summary Tables of Changes*.

Q37. Incorrect Access Controls to MSR_LASTBRANCH_0_FROM_LIP MSR Registers

Problem: When an access is made to the MSR_LASTBRANCH_0_FROM_LIP MSR register, an expected #GP fault may not happen.

Implication: A read of the MSR_LASTBRANCH_0_FROM_LIP MSR register may not cause a #GP fault.

Workaround: It is possible for the BIOS to contain a workaround for this erratum.

Status: For the steppings affected, see the *Summary Tables of Changes*.

Q38. Recursive Page Walks May Cause a System Hang

Problem: A page walk, accessing the same page table entry multiple times but at different levels of the page table, which causes the page table entry to have its Access bit set may result in a system hang.

Implication: When this erratum occurs, the system may experience a hang.

Workaround: It is possible for the BIOS to contain a workaround for this erratum.

Status: For the steppings affected, see the *Summary Tables of Changes*.

Q39. WRMSR to bit[0] of IA32_MISC_ENABLE Register Changes Only One Logical Processor on a Hyper-Threading Technology Enabled Processor

Problem: On an HT enabled processor, a write to the fast-strings feature bit[0] of IA32_MISC_ENABLE register changes the setting for the current logical processor only.

Implication: Due to this erratum, the non-current logical processor may not update fast-strings feature bit[0] of IA32_MISC_ENABLE register.

Workaround: BIOS may set the fast-strings enable bit on both logical processors to workaround this erratum. It is possible for the BIOS to contain a workaround for this erratum.

Status: For the steppings affected, see the *Summary Tables of Changes.*

Q40. Data Breakpoints on the High Half of a Floating Point Line Split May Not Be Captured

Problem: When a floating point load which splits a 64-byte cache line gets a floating point stack fault, and a data breakpoint register maps to the
high line of the floating point load, internal boundary conditions exist that may prevent the data breakpoint from being captured.

Implication: When this erratum occurs, a data breakpoint will not be captured.

Workaround: None identified.

Status: For the steppings affected, see the *Summary Tables of Changes.*

Q41. Machine Check Exceptions May Not Update Last-Exception Record MSRs (LERs)

Problem: The Last-Exception Record MSRs (LERs) may not get updated when Machine Check Exceptions occur.

Implication: When this erratum occurs, the LER may not contain information relating to the machine check exception. They will contain information relating to the exception prior to the machine check exception.

Workaround: None identified.

Status: For the steppings affected, see the *Summary Tables of Changes.*

Q42. MOV CR3 Performs Incorrect Reserved Bit Checking When in PAE Paging

Problem: The MOV CR3 instruction should perform reserved bit checking on the upper unimplemented address bits. This checking range should match the address width reported by CPUID instruction 0x8000008. This erratum applies whenever PAE is enabled.

Implication: Software that sets the upper address bits on a MOV CR3 instruction and expects a fault may fail. This erratum has not been observed with commercially available software.

Workaround: None identified.

Status: For the steppings affected, see the *Summary Tables of Changes.*

Q43. Stores to Page Tables May Not Be Visible to Pagewalks for Subsequent Loads Without Serializing or Invalidating the Page Table Entry

Problem: Under rare timing circumstances, a page table load on behalf of a programmatically younger memory access may not get data from a programmatically older store to the page table entry if there is not a fencing operation or page translation invalidate operation between the store and the younger memory access. Refer to the IA-32 Intel® Architecture Software Developer's Manual for the correct way to update page tables. Software that conforms to the Software Developer's Manual will operate correctly.

Implication: If the guidelines in the Software Developer's Manual are not followed, stale data may be loaded into the processor's Translation Lookaside Buffer (TLB) and used for memory operations. This erratum has not been observed with any commercially available software.

Workaround: The guidelines in the IA-32 Intel® Architecture Software Developer's Manual should be followed.

Status: For the steppings affected, see the *Summary Tables of Changes.*

Q44. A Split Store Memory Access May Miss a Data Breakpoint

Problem: It is possible for a data breakpoint specified by a linear address to be missed during a split store memory access. The problem can happen with or without paging enabled.

Implication: This erratum may limit the debug capability of debugger software.

Workaround: None identified.

Status: For the steppings affected, see the *Summary Tables of Changes.*

Q45. Execution of IRET or INTn Instructions May Cause Unexpected System Behavior

Problem: There is a small window of time, requiring alignment of many internal micro architectural events, during which the speculative execution of the IRET or INTn instructions in protected or IA-32e mode may result in unexpected software or system behavior.

Implication: This erratum may result in unexpected instruction execution, events, interrupts or a system hang when the IRET instruction is executed. The execution of the INTn instruction may cause debug breakpoints to be missed.

Workaround: It is possible for the BIOS to contain a workaround for this erratum.

Status: For the steppings affected, see the *Summary Tables of Changes.*

Q46. Checking of Page Table Base Address May Not Match the Address Bit Width Supported by the Platform

Problem: If the page table base address, included in the page map level-4 table, page-directory pointer table, page-directory table or page table, exceeds the physical address range supported by the platform (e.g. 36-bit) and it is less than the implemented address range (e.g. 40-bit), the processor does not check if the address is invalid.

Implication: If software sets such invalid physical address in those tables, the processor does not generate a page fault (#PF) upon access to that virtual address, and the access results in an incorrect read or write. If BIOS provides only valid physical address ranges to the operating system, this erratum will not occur.

Workaround: BIOS must provide valid physical address ranges to the operating system.

Status: For the steppings affected, see the *Summary Tables of Changes*.

Q47. The IA32_MCi_STATUS MSR May Improperly Indicate That Additional MCA Information May Have Been Captured

Problem: When a data parity error is detected and the bus queue is busy, the ADDRV and MISCV bits of the IA32_MCi_STATUS register may be asserted even though the contents of the IA32_MCi_ADDR and IA32_MCi_MISC MSRs were not properly captured.

Implication: If this erratum occurs, the MCA information captured in the IA32_MCi_ADDR and IA32_MCi_MISC may not correspond to the reported machine-check error, even though the ADDRV and MISCV are asserted.

Workaround: None identified.

Status: For the steppings affected, see the *Summary Tables of Changes*.

Q48. With TF (Trap Flag) Asserted, FP Instruction That Triggers an Unmasked FP Exception May Take Single Step Trap Before Retirement of Instruction

Problem: If an FP instruction generates an unmasked exception with the EFLAGS.TF=1, it is possible for external events to occur, including a transition to a lower power state. When resuming from the lower power state, it may be possible to take the single step trap before the execution of the original FP instruction completes.

Implication: A Single Step trap will be taken when not expected.

Workaround: None identified.

Status: For the steppings affected, see the *Summary Tables of Changes*.

Q49. MCA Corrected Memory Hierarchy Error Counter May Not Increment Correctly

Problem: An MCA corrected memory hierarchy error counter can report a maximum of 255 errors. Due to the incorrect increment of the counter, the number of errors reported may be incorrect.

Implication: Due to this erratum, the MCA counter may report incorrect number of soft errors.

Workaround: None Identified

Status: For the steppings affected, see the *Summary Tables of Changes*.

Q50. BTS(Branch Trace Store) and PEBS(Precise Event Based Sampling) May Update Memory outside the BTS/PEBS Buffer

Problem: If the BTS/PEBS buffer is defined such that:

- The difference between BTS/PEBS buffer base and BTS/PEBS absolute maximum is not an integer multiple of the corresponding record sizes
- BTS/PEBS absolute maximum is less than a record size from the end of the virtual address space
- The record that would cross BTS/PEBS absolute maximum will also continue past the end of the virtual address space

Implication: Software that uses BTS/PEBS near the 4G boundary (IA32) or 2^64 boundary (EMT64T mode), and defines the buffer such that it does not hold an integer multiple of records can update memory outside the BTS/PEBS buffer.

Workaround: Define BTS/PEBS buffer such that BTS/PEBS absolute maximum minus BTS/PEBS buffer base is integer multiple of the corresponding record sizes as recommended in the IA-32 Intel® Architecture Software Developer's Manual Volume 3.

Status: For the steppings affected, see the *Summary Tables of Changes.*

Q51. Processor May Hang When Resuming from Deep Sleep State

Problem: When resuming from the Deep Sleep state, the IO Phase Lock Loop (IOPLL) may not properly lock.

Implication: When this erratum occurs, the processor may hang.

Workaround: The system BIOS should prevent the processor from going to the Deep Sleep state.

Status: For the steppings affected, see the *Summary Tables of Changes*.

Q52. Memory Ordering Failure May Occur with Snoop Filtering Third- Party Agents after Issuing and Completing a BWIL (Bus Write Invalidate Line) or BLW (Bus Locked Write) Transaction

Problem: Under limited circumstances, the processors may, after issuing and completing a BWIL or BLW transaction, retain data from the addressed cache line in shared state even though the specification requires complete invalidation. This data retention may also occur when a BWIL transaction's self-snooping yields HITM snoop results.

Implication: A system may suffer memory ordering failures if its central agent incorporates coherence sequencing which depends on full self-invalidation of the cache line associated with (1) BWIL and BLW transactions, or (2) all HITM snoop results without regard to the transaction type and snoop results' source.

Workaround: 1. The central agent can issue a bus cycle that causes a cache line to be invalidated (Bus Read Invalidate Line (BRIL) or BWIL transaction) in response to a processor-generated BWIL (or BLW) transaction to insure complete invalidation of the associated cache line. If there are no intervening processor-originated transactions to that cache line, the central agent's invalidating snoop will get a clean snoop result.

Or

2. Snoop filtering central agents can:
   a. Not use processor-originated BWIL or BLW transactions to update their snoop filter information, or
   b. Update the associated cache line state information to shared state on the originating bus (rather than invalid state) in reaction to a BWIL or BLW.

Status: For the steppings affected, see the *Summary Tables of Changes*.

Q53. Control Register 2 (CR2) Can be Updated during a REP MOVS/STOS Instruction with Fast Strings Enabled

Problem: Under limited circumstances while executing a REP MOVS/STOS string instruction, with fast strings enabled, it is possible for the value in CR2 to be changed as a result of an interim paging event, normally invisible to the user. Any higher priority architectural event that arrives and is handled while the interim paging event is occurring may see the modified value of CR2.

Implication: The value in CR2 is correct at the time that an architectural page fault is signaled. Intel has not observed this erratum with any commercially available software.

Workaround: None Identified

Status: For the steppings affected, see the *Summary Tables of Changes*.

Q54. TPR (Task Priority Register) Updates during Voltage Transitions of Power Management Events May Cause a System Hang

Problem: Systems with Echo TPR Disable (R/W) bit (bit [23] of IA32_MISC_ENABLE register) set to '0' (default), where xTPR messages are being transmitted on the system bus to the processor, may experience a system hang during voltage transitions caused by the power management events.

Implication: This may cause a system hang during voltage transitions of power management events.

Workaround: It is possible for the BIOS to contain a workaround for this erratum. The BIOS workaround disables the Echo TPR updates on affected steppings.

Status: For the steppings affected, see the *Summary Tables of Changes*.

Q55. Running in SMM (System Management Mode) and L1 Data Cache Adaptive Mode May Cause Unexpected System Behavior when SMRAM is Mapped to Cacheable Memory

Problem: In a Hyper-Threading Technology-enabled system, unexpected system behavior may occur if a change is made to the value of the CR3 result from an RSM (Resume From System Management) instruction while in L1 data cache adaptive mode (IA32_MISC_ENABLES MSR 0x1a0 bit 24). This behavior will only be visible when SMRAM is mapped into WB/WT cacheable memory on SMM entry and exit.

Implication: This erratum can have multiple failure symptoms including incorrect data in memory. Intel has not observed this erratum with any commercially available software.

Workaround: Disable L1 data cache adaptive mode by setting the L1 Data Cache Context Mode control (bit 24) of the IA32_MISC_ENABLES MSR (0x1a0) to 1.

Status: For the steppings affected, see the *Summary of Table of Changes*

Q56. Voltage and Frequency Transition May Not Occur if a Voltage Transition is Interrupted by a Warm Reset

Problem: In an Enhanced Intel SpeedStep® technology or Thermal Monitor 2 enabled system, if a voltage and frequency transition is interrupted by a warm reset and the next transition is to a higher voltage and frequency then that transition and all subsequent transitions will not be performed.

Implication: When this erratum occurs, the processor will not perform any further transitions and will remain at the Reset# voltage and frequency. Intel has not observed this erratum with any commercially available system.

Workaround: None identified.

Status: For the steppings affected, see the *Summary Tables of Changes*.

Q57. It Is Possible That Two Specific Invalid Opcodes May Cause Unexpected Memory Accesses

Problem: A processor is expected to respond with an undefined opcode (#UD) fault when executing either opcode 0F 78 or a Grp 6 Opcode with bits 5:3 of the Mod/RM field set to 6, however the processor may respond instead, with a load to an incorrect address.

Implication: This erratum may cause unpredictable system behavior or system hang.

Workaround: It is possible for BIOS to contain a workaround for this erratum.

Status: For the steppings affected, see the *Summary of Tables of Changes.*

Q58. At Core-to-Bus Ratios of 16:1 and above Defer Reply Transactions with Non-Zero REQb Values May Cause a Front Side Bus Stall

Problem: Problem: Certain processors are likely to hang the Front Side Bus (FSB) if the following conditions are met:

1. A Defer Reply transaction has a REQb[2:0] value of either 010b, 011b, 100b, 110b, or 111b, and
2. The operating bus ratio is 16:1 or higher

When these conditions are met, the processor may incorrectly and indefinitely assert a snoop stall for the Defer Reply transaction. Such an event will block further progress on the FSB.

Implication: If this erratum occurs, the system may hang. Intel has not observed this erratum with any commercially available system.

Workaround: None identified.

Status: For the steppings affected, see the *Summary of Tables of Changes.*

Q59. The Processor May Issue Front Side Bus Transactions up to 6 Clocks after RESET# is Asserted

Problem: The processor may issue transactions beyond the documented 3 Front Side Bus (FSB) clocks and up to 6 FSB clocks after RESET# is asserted in the case of a warm reset. A warm reset is where the chipset asserts RESET# when the system is running.

Implication: The processor may issue transactions up to 6 FSB clocks after RESET# is asserted.

Workaround: None identified.

Status: For the steppings affected, see the *Summary of Tables of Changes.*

Q60. Front Side Bus Machine Checks May be Reported as a Result of On-Going Transactions during Warm Reset

Problem: Processor Front Side Bus (FSB) protocol/signal integrity machine checks may be reported if the transactions are initiated or in-progress during a warm reset. A warm reset is where the chipset asserts RESET# when the system is running.

Implication: The processor may log FSB protocol/signal integrity machine checks if transactions are allowed to occur during RESET# assertions.

Workaround: BIOS may clear FSB protocol/signal integrity machine checks for systems/chipsets which do not block new transactions during RESET# assertions.

Status: For the steppings affected, see the *Summary of Tables of Changes.*

Q61. At Core-to-Bus Ratios of 16:1 and above Defer Reply Transactions with Non-Zero REQb Values May Cause a Front Side Bus Stall

Problem: If a local interrupt is pending when the LVT entry is written, an interrupt may be taken on the new interrupt vector even if the mask bit is set.

Implication: An interrupt may immediately be generated with the new vector when a LVT entry is written, even if the new LVT entry has the mask bit set. If there is no Interrupt Service Routine (ISR) set up for that vector the system will GP fault. If the ISR does not do an End of Interrupt (EOI) the bit for the vector will be left set in the in-service register and mask all interrupts at the same or lower priority.

Workaround: Any vector programmed into an LVT entry must have an ISR associated with it, even if that vector was programmed as masked. This ISR routine must do an EOI to clear any unexpected interrupts that may occur. The ISR associated with the spurious vector does not generate an EOI, therefore the spurious vector should not be used when writing the LVT.

Status: For the steppings affected, see the *Summary Tables of Changes*.

Q62. The Processor May Issue Multiple Code Fetches to the Same Cache Line for Systems with Slow Memory

Problem: Systems with long latencies on returning code fetch data from memory e.g. BIOS ROM, may cause the processor to issue multiple fetches to the same cache line, once per each instruction executed.

Implication: This erratum may slow down system boot time. Intel has not observed a failure, as a result of this erratum, in a commercially available system.

Workaround: None identified.

Status: For the steppings affected, see the *Summary Tables of Changes*.

§

Specification Changes

The Specification Changes listed in this section apply to the following documents:

- *Mobile Intel® Pentium® 4 Processor supporting Hyper-Threading Technology on 90-nm process technology Datasheet*

All Specification Changes will be incorporated into a future version of the appropriate Mobile Pentium 4 Processor supporting Hyper-Threading Technology on 90-nm process technology documentation.

Q1. Update to Processor T73: Deep Sleep PLL Lock Latency

Deep Sleep PLL Lock Latency for CPUID = 0xF41 has changed from 30 µs to 245 µs for the Max specification.

Deep Sleep State—State 6

Deep Sleep state is a very low power state the processor can enter while maintaining context. Deep Sleep state is entered by asserting the DPSLP# pin while in the Sleep state. The DPSLP# pin must be de-asserted to re-enter the Sleep state. A period of 245 microseconds (to allow for PLL stabilization) must occur before the processor can be considered to be in the Sleep State. Once in the Sleep state, the SLP# pin can be deasserted to re-enter the Stop-Grant state.

§

Specification Clarifications

All Specification Clarifications will be incorporated into a future version of the appropriate Mobile Pentium 4 processor documentation.

Q1. Specification Clarification with Respect to Time-stamp Counter

In the "Debugging and Performance Monitoring" section (Sections 15.8, 15.10.9 and 15.10.9.3) of the *IA-32 Intel® Architecture Software Developer's Manual Volume 3: System Programming Guide*, the Time-stamp Counter definition has been updated to include support for the future processors. This change will be incorporated in the next revision of the *IA-32 Intel® Architecture Software Developer's Manual*.

15.8 Time-stamp Counter

The IA-32 architecture (beginning with the Pentium processor) defines a time-stamp counter mechanism that can be used to monitor and identify the relative time occurrence of processor events. The counter's architecture includes the following components:

- **TSC flag —** A feature bit that indicates the availability of the time-stamp counter. The counter is available in an IA-32 processor implementation if the function CPUID.1:EDX.TSC[bit 4] = 1.
- **IA32_TIME_STAMP_COUNTER MSR** (called TSC MSR in P6 family and Pentium processors) **—** The MSR used as the counter.
- **RDTSC instruction —** An instruction used to read the time-stamp counter.
- **TSD flag —** A control register flag is used to enable or disable the time-stamp counter (enabled if CR4.TSD[bit 2] = 1).

The time-stamp counter (as implemented in the P6 family, Pentium, Pentium M, Pentium 4, and Intel Xeon processors) is a 64-bit counter that is set to 0 following a RESET of the processor. Following a RESET, the counter will increment even when the processor is halted by the HLT instruction or the external STPCLK# pin. Note that the assertion of the external DPSLP# pin may cause the time-stamp counter to stop.

Members of the processor families increment the time-stamp counter differently:

- For Pentium M processors (family [06H], models [09H, 0DH]); for Pentium 4 processors, Intel Xeon processors (family [0FH], models [00H, 01H, or 02H]); and for P6 family processors: the time-stamp counter increments with every internal processor clock cycle. The internal processor clock cycle is determined by the current core-clock to bus-clock ratio. Intel SpeedStep® technology transitions may also impact the processor clock.
- For Pentium 4 processors, Intel Xeon processors (family [0FH], models [03H and higher]): the time-stamp counter increments at a constant rate. That rate may be set by the maximum core-clock to bus-clock ratio of the processor or may be set by the frequency at which the processor is booted. The specific processor

configuration determines the behavior. Constant TSC behavior ensures that the duration of each clock tick is uniform and supports the use of the TSC as a wall clock timer even if the processor core changes frequency. This is the architectural behavior moving forward.

Note: To determine average processor clock frequency, Intel recommends the use of Performance Monitoring logic to count processor core clocks over the period of time for which the average is required. See Section 15.10.9 and Appendix A in this manual for more information.

The RDTSC instruction reads the time-stamp counter and is guaranteed to return a monotonically increasing unique value whenever executed, except for a 64-bit counter wraparound. Intel guarantees that the time-stamp counter will not wraparound within 10 years after being reset. The period for counter wrap is longer for Pentium 4, Intel Xeon, P6 family, and Pentium processors.

Normally, the RDTSC instruction can be executed by programs and procedures running at any privilege level and in virtual-8086 mode. The TSD flag allows use of this instruction to be restricted to programs and procedures running at privilege level 0. A secure operating system would set the TSD flag during system initialization to disable user access to the time-stamp counter. An operating system that disables user access to the time-stamp counter should emulate the instruction through a user-accessible programming interface.

The RDTSC instruction is not serializing or ordered with other instructions. It does not necessarily wait until all previous instructions have been executed before reading the counter. Similarly, subsequent instructions may begin execution before the RDTSC instruction operation is performed.

The RDMSR and WRMSR instructions read and write the time-stamp counter, treating the time-stamp counter as an ordinary MSR (address 10H). In the Pentium 4, Intel Xeon, and P6 family processors, all 64-bits of the time-stamp counter are read using RDMSR (just as with RDTSC). When WRMSR is used to write the time-stamp counter on processors before family [0FH], models [03H, 04H]: only the low order 32-bits of the time-stamp counter can be written (the high-order 32 bits are cleared to 0). For family [0FH], models [03H, 04H]: all 64 bits are writeable.

15.10.9 Counting Clocks

The count of cycles, also known as clockticks, forms the basis for measuring how long a program takes to execute. Clockticks are also used as part of efficiency ratios like cycles per instruction (CPI). Processor clocks may stop ticking under circumstances like the following:

- The processor is halted when there is nothing for the CPU to do. For example, the processor may halt to save power while the computer is servicing an I/O request. When Hyper-Threading Technology is enabled, both logical processors must be halted for performance-monitoring counters to be powered down.
- The processor is asleep as a result of being halted or because of a power-management scheme. There are different levels of sleep. In the some deep sleep levels, the time-stamp counter stops counting.

There are three ways to count processor clock cycles to monitor performance. These are:

- **Non-halted clockticks —** Measures clock cycles in which the specified logical processor is not halted and is not in any power-saving state. When Hyper-Threading Technology is enabled, these ticks can be measured on a per-logical-processor basis.
- **Non-sleep clockticks —** Measures clock cycles in which the specified physical processor is not in a sleep mode or in a power-saving state. These ticks cannot be measured on a logical-processor basis.

- **Time-stamp counter** — Some processor models permit clock cycles to be measured when the physical processor is not in deep sleep (by using the time-stamp counter and the RDTSC instruction). Note that such ticks cannot be measured on a per-logical-processor basis. See Section 10.8 for detail on processor capabilities.

The first two methods use performance counters and can be set up to cause an interrupt upon overflow (for sampling). They may also be useful where it is easier for a tool to read a performance counter than to use a time-stamp counter (the timestamp counter is accessed using the RDTSC instruction).

For applications with a significant amount of I/O, there are two ratios of interest:

- **Non-halted CPI —** Non-halted clockticks/instructions retired measures the CPI for phases where the CPU was being used. This ratio can be measured on a logical-processor basis when Hyper-Threading Technology is enabled.
- **Nominal CPI —** Time-stamp counter ticks/instructions retired measures the CPI over the duration of a program, including those periods when the machine halts while waiting for I/O.

15.10.9.3 Incrementing the Time-Stamp Counter

The time-stamp counter increments when the clock signal on the system bus is active and when the sleep pin is not asserted. The counter value can be read with the RDTSC instruction.
The time-stamp counter and the non-sleep clockticks count may not agree in all cases and for all processors. See Section 10.8 for more information on counter operation.

§

Documentation Changes

The Documentation Changes listed in this section apply to the following documents:

- *Mobile Intel® Pentium® 4 Processor supporting Hyper-Threading Technology on 90-nm process technology Datasheet*

All Documentation Changes will be incorporated into a future version of the appropriate Mobile Pentium 4 processor documentation.

Note: Documentation changes for IA-32 Intel® Architecture Software Developer's Manual volumes 1 - 3 will be posted in a separate document, *IA-32 Intel® Architecture Software Developer's Manual Documentation Changes*. Follow the link below to become familiar with this file.

http://developer.intel.com/design/pentium4/specupdt/252046.htm

There are no documentation changes in this Specification Update revision.

§